BAB II
LANDASAN TEORI

2.1 Pengertian Data Mining

Data mining adalah serangkaian proses untuk menggali nilai tambah berupa informasi yang selama ini tidak diketahui secara manual dari suatu basis data. Informasi yang dihasilkan diperoleh dengan cara mengekstraksi dan mengenali pola yang penting atau menarik dari data yang terdapat dalam basis data. Data mining adalah proses yang menggunakan teknik statistik, matematika, kecerdasan buatan, dan machine learning untuk mengekstraksi dan mengidentifikasi informasi yang bermanfaat dan pengetahuan yang terkait dari berbagai database besar.

Menurut Gartner Group data mining adalah suatu proses menemukan hubungan yang berarti, pola, dan kecenderungan dengan memeriksa dalam sekumpulan besar data yang tersimpan dalam penyimpanan dengan menggunakan teknik pengenalan pola seperti teknik statistik dan matematika. Data mining bukanlah suatu bidang yang sama sekali baru. Salah satu kesulitan untuk mendefinisikan data mining adalah kenyataan bahwa data mining mewarisi banyak aspek dan teknik dari bidang-bidang ilmu yang sudah mapan terlebih dulu. Berawal dari beberapa disiplin ilmu, data mining bertujuan untuk memperbaiki teknik tradisional sehingga bisa menangani:

- Jumlah data yang sangat besar
- Dimensi data yang tinggi
- Data yang heterogen dan berbeda sifat

Menurut para ahli, data mining merupakan sebuah analisa dari observasi data dalam jumlah besar untuk menemukan hubungan yang tidak diketahui sebelumnya dan metode baru untuk meringkas data agar mudah dipahami serta kegunaannya untuk pemilik data.

Data-data yang ada, tidak dapat langsung diolah dengan menggunakan sistem data mining. Data-data tersebut harus dipersiapkan terlebih dahulu agar

hasil yang diperoleh dapat lebih maksimal, dan waktu komputasinya lebih minimal. Proses persiapan data ini sendiri dapat mencapai 60 % dari keseluruhan proses dalam data mining. Karena itu *data mining* sebenarnya memiliki akar yang panjang dari bidang ilmu seperti kecerdasan buatan (*artificial intelligent*), *machine learning*, statistik dan database. Beberapa metode yang sering disebut-sebut dalam literatur *data mining* antara lain *clustering*, *yesification*, *association rules mining*, *neural network*, *genetic algorithm* dan lain-lain (Pramudiono, 2007).

2.2 Tahap-tahap Data Mining

Sebagai suatu rangkaian proses, *data mining* dapat dibagi menjadi beberapa tahap yang diilustrasikan di Gambar, Tahap-tahap tersebut bersifat interaktif, pemakai terlibat langsung dengan perantaraan *knowledge base*.

Tahap-tahap *data mining* ada 6 yaitu :

1. Pembersihan data (*data cleaning*).

   Pembersihan data merupakan proses menghilangkan *noise* dan data yang tidak konsisten atau data yang tidak relevan. Pada umumnya data diperoleh, baik dari *database* suatu perusahaan maupun hasil eksperimen, memiliki isian-isian yang tidak sempurna seperti data yang hilang, data yang tidak valid atau juga hanya sekedar salah ketik. Selain itu, ada juga atribut-atribut data yang tidak relevan dengan hipotesa *data mining* yang dimiliki. Data-data yang tidak relevan itu juga lebih baik dibuang. Pembersihan data juga akan mempengaruhi performansi dari teknik *data mining* karena data yang ditangani akan berkurang jumlah dan kompleksitasnya.

2. Integrasi Data (*data integraton*)

   Integrasi data merupakan penggabungan data dari berbagai *database* ke dalam satu *database* baru. Tidak jarang data yang diperlukan untuk *data mining* tidak hanya berasal dari satu *database* atau file teks. Integrasi data dilakukan pada atribut-atribut yang mengidentifikasikan entitas-entitas yang unik seperti atribut nama, jenis produk, nomor pelanggan dan lainnya. Integrasi data perlu dilakukan secara cermat karena

kesalahan pada integrasi data bisa menghasilkan hasil yang menyimpang dan bahkan menyesatkan pengambilan aksi nantinya. Sebagai contoh bila integrasi data berdasarkan jenis produk dari kategori yang berbeda maka akan didapatkan korelasi antar produk yang sebenarnya tidak ada.

3. Seleksi Data (*Data Selection*)

   Data yang ada pada *database* sering kali tidak semuanya dipakai, oleh karena itu hanya data yang sesuai untuk dianalisis yang akan diambil dari *database*. Sebagai contoh, sebuah kasus yang meneliti faktor kecendrungan orang membeli dalam kasus *market analysis*, tidak perlu mengambil nama pelanggan, cukup dengan id pelanggan saja.

4. Transformasi Data (*data transformation*)

   Data diubah atau digabung ke dalam format yang sesuai untuk diproses dalam *data mining*. Beberapa metode *data mining* membutuhkan format data yang khusus sebelum bisa diaplikasikan. Sebagai contoh beberapa metode standar seperti analisis asoisasi dan *clustering* hanya bisa menerima input data kategorikal, Karenanya data berupa angka numerik yang berlanjut perlu dibagi-bagi menjadi beberapa interval. Proses ini sering disebut transformasi data.

5. Proses *Mining*

   Merupakan suatu proses utama saat metode diterapkan untuk menemukan pengetahuan berharga dan tersembunyi dari data.

6. Evaluasi Pola (*pattern evaluation*).

   Untuk mengidentifikasi pola-pola menarik kedalam *knowledge based* yang ditemukan. Dalam tahap ini hasil dari teknik *data mining* berupa pola-pola yang khas maupun model prediksi dievaluasi untuk menilai apakah hipotesa yang ada tercapai. Bila ternyata hasil yang diperoleh tidak sesuai hipotesa ada beberapa alternatif yang dapat diambil seperti menjadikannya umpan balik untuk memperbaiki proses *data mining*, mencoba metode *data mining* yang lebih sesuai, atau menerima hasil ini sebagai suatu hal yang diluar dugaan yang mungkin bermanfaat.

2.3 Pengelompokan Data Mining

Data mining dibagi menjadi beberapa kelompok berdasarkan tugas yang dapat dilakukan, yaitu (Daniel T. Larose, 2005):

1. Deskripsi

   Deskripsi adalah menggambaran pola dan kecenderungan yang terdapat dalam data secara sederhana. Deskripsi dari pola dan kecenderungan sering memberikan kemungkinan penjelasan untuk suatu pola atau kecenderungan.

2. Klasifikasi

   Suatu teknik dengan melihat pada kelakuan dan atribut dari kelompok yang telah didefinisikan. Teknik ini dapat memberikan klasifikasi pada data baru dengan memanipulasi data yang telah diklasifikasi dan dengan menggunakan hasilnya untuk memberikan sejumlah aturan. Klasifikasi menggunakan *supervisedlearning*.

3. Estimasi

   Estimasi hampir sama dengan klasifikasi, perbedaanya adalah variabel target estimasi lebih ke arah numerik daripada ke arah kategori. Model dibangun dengan menggunakan *record* lengkap yang menyediakan nilai dari variabel target sebagai nilai prediksi.

4. Prediksi

   Prediksi memiliki kesamaan dengan klasifikasi dan estimasi, perbedaanya adalah hasil dari prediksi akan ada dimasa mendatang. Beberapa teknik yang digunakan dalam klasifikasi dan estimasi dapat juga digunakan (untuk keadaan yang tepat) untuk prediksi.

5. Klastering

   Klastering merupakan pengelompokan *record*, pengamatan, atau memperhatikan dan membentuk kelas objek-obek yang memiliki kemiripan satu dengan yang lainnya dan memiliki ketidakmiripan dengan *record-record* dalam kluster lain. Klastering menggunakan *unsupervised learning*.

6. Asosiasi

   Tugas asosiasi atau sering disebut juga sebagai *market basket analysis* dalam data mining adalah menemukan relasi atau korelasi diantara himpunan item-item dan menemukan atribut yang muncul dalam satu waktu. Asosiasi menggunakan *unsupervised learning*. Penting tidaknya suatu aturan assosiatif dapat diketahui dengan dua parameter, *support* dan *confidence*.

2.4 Decision Tree

Decision tree merupakan metode klasifikasi *data mining*. *Decision tree* dalam istilah pembelajaran merupakan sebuah struktur pohon dimana setiap *node* pohon mempresentasikan atribut yang telah diuji. Setiap cabang merupakan suatu pembagian hasil uji dan node daun (*leaf*) mempresentasikan kelompok kelas tertentu. (Jianwei, Han. 2001). Level *node* teratas dari sebuah *Decision Tree* adalah node akar (root) yang biasanya berupa atribut yang paling memiliki pengaruh terbesar pada suatu kelas tertentu. Pada umumnya *Decision Tree* melakukan strategi pencarian secara top-down untuk solusinya. Pada proses mengklasifikasi data yang tidak diketahui, nilai atribut akan diuji dengan cara melacak jalur dari *node* akar (*root*) sampai *node* akhir (daun) dan kemudian akan diprediksi kelas yang dimiliki oleh suatu data baru tertentu. (Santosa, Budi. 2007).

2.4.1 Jenis-Jenis *Decision Tree*

Beberapa model *decision tree* yang sudah dikembangkan antara lain C4.5 atau ID3 dan CART. Berikut ini akan dijelaskan model dari decision tree tersebut:

1. C4.5 atau ID3

   Decision Tree menggunakan algoritma ID3 atau C4.5, yang diperkenalkan dan dikembangkan pertama kali oleh Quinlan yang merupakan singkatan dari *Iterative Dichotomiser 3* atau *Induction of Decision 3.* Algoritma ID3 membentuk pohon keputusan dengan metode divide-and-conquer data secara rekursif dari atas ke bawah. Strategi pembentukan Decision Tree dengan algoritma ID3 adalah:

A. Pohon dimulai sebagai node tunggal (akar/root) yang merepresentasikan semua data.
B. Sesudah node root dibentuk, maka data pada node akar akan diukur dengan information gain untuk dipilih atribut mana yang akan dijadikan atribut pembaginya.
C. Sebuah cabang dibentuk dari atribut yang dipilih menjadi pembagi dan data akan didistribusikan ke dalam cabang masing-masing.
D. Algoritma ini akan terus menggunakan proses yang sama atau bersifat rekursif untuk dapat membentuk sebuah Decision Tree. Ketika sebuah atribut telah dipilih menjadi node pembagi atau cabang, maka atribut tersebut tidak diikutkan lagi dalam penghitungan nilai information gain.
E. Proses pembagian rekursif akan berhenti jika salah satu dari kondisi dibawah ini terpenuhi :
   a. Semua data dari anak cabang telah termasuk dalam kelas yang sama.
   b. Semua atribut telah dipakai, tetapi masih tersisa data dalam kelas yang berbeda. Dalam kasus ini, diambil data yang mewakili kelas yang terbanyak untuk menjadi label kelas pada node daun. Tidak terdapat data pada anak cabang yang baru. Dalam kasus ini, node daun akan dipilih pada cabang sebelumnya dan diambil data yang mewakili kelas terbanyak untuk dijadikan label kelas.

Metode C4.5 dan ID3 memliki perbedaan dalam nilai tiap atribut. Metode C4.5 menggunakan atribut yang bernilai kategorikal dan numerikal, sedangkan metode ID3 menggunakan atribut yang bernilai kategorikal. Metode *decision tree ID3* inilah yang digunakan dalam tugas akhir ini.

2. CART

CART adalah singkatan dari *Yesification And Regression Tree*. Dalam CART ada dua langkah penting yang harus diikuti untuk mendapatkan *tree* dengan performansi yang optimal. Yang pertama adalah pemecahan objek secara berulang berdasarkan atribut tertentu. Yang kedua, *prunning* (pemangkasan) dengan menggunakan data validasi.

Misalkan kita mempunyai variabel independent $x_1,x_2,x_3,.....x_n$ dan variabel dependent atau output y. Pemecahan secara berulang berarti kita bagi objek ke dalam kotak-kotak berdasarkan nilai variabel x_1,x_2 atau x_r. Cara ini diulang sehingga dalam suatu kotak sebisa mungkin berisi observasi dalam kelompok atau kelas yang sama.

Langkah berikutnya sesudah dilakukan pemecahan objek atau data secara berulang adalah melakukan *prunning*. Dalam *prunning* kita ingin memangkas *tree* yang mungkin terlalu besar dan terjadi fenomena *overfitting*. *Overvitting* merupakan sebuah satu buah pengelompokkan yang mungkin hanya berisi satu data yang memungkinkan data tersebut merupakan *noise* yang ada di data training dan bukan pola yang mungkin terjadi dalam data testing atau data validasi. *Prunning* terdiri dari beberapa langkah pemilihan secara berulang simpul yang akan dijadikan simpul daun. Dengan mengubah simpul menjadi simpul daun artinya tidak akan dilakukan pemecahan lagi sesudah itu. Dengan demikian ukuran *tree* akan berkurang. (Santosa, Budi. 2007).

2.4.2 Pohon keputusan

Pohon Keputusan *(Decision tree)* adalah salah satu metode yang sangat populer dan banyak digunakan secara praktis. Metode ini merupakan metode yang berusaha menemukan fungsi-fungsi pendekatan yang bernilai diskrit dan tahan terhadap data yang terdapat kesalahan (*noisy data*) serta mampu mempelajari ekspresi-ekspresi *disjunctive* (ekspresi *OR*).

Decision tree (pohon keputusan) adalah sebuah diagram alir yang mirip dengan struktur pohon, dimana setiap *internal node* menotasikan atribut yang diuji, setiap cabangnya mempresentasikan hasil dari atribut tes tersebut dan *leaf node* mepresentasikan kelas-kelas tertentu atau distribusi dari kelas-kelas (Han,2001).

Alur pada *decision tree* ditelusuri dari simpul ke akar ke simpul daun yang memegang prediksi kelas untuk contoh tersebut. *decision tree* mudah untuk

dikonversi ke aturan klasifikasi (*yesification rule*). Konsep data dalam *decision tree* dinyatakan dalam bentuk tabel dengan atribut dan *record*.

Manfaat utama dari penggunaan pohon keputusan adalah kemampuannya untuk mem-*break down* proses pengambilan keputusan yang kompleks menjadi lebih simpel sehingga pengambil keputusan akan lebih menginterpretasikan solusi dari permasalahan. Pohon Keputusan juga berguna untuk mengeksplorasi data, menemukan hubungan tersembunyi antara sejumlah calon variabel input dengan sebuah variabel target. Pohon keputusan memadukan antara eksplorasi data dan pemodelan, sehingga sangat bagus sebagai langkah awal dalam proses pemodelan bahkan ketika dijadikan sebagai model akhir dari beberapa teknik lain. Sering terjadi tawar menawar antara keakuratan model dengan transparansi model. Dalam beberapa aplikasi, akurasi dari sebuah klasifikasi atau prediksi adalah satu-satunya hal yang ditonjolkan, misalnya sebuah perusahaan *direct mail* membuat sebuah model yang akurat untuk memprediksi anggota mana yang berpotensi untuk merespon permintaan, tanpa memperhatikan bagaimana atau mengapa model tersebut bekerja.

Kelebihan dari metode pohon keputusan adalah:

1. Daerah pengambilan keputusan yang sebelumnya kompleks dan sangat global, dapat diubah menjadi lebih simpel dan spesifik.
2. Eliminasi perhitungan-perhitungan yang tidak diperlukan, karena ketika menggunakan metode pohon keputusan maka *sample* diuji hanya berdasarkan kriteria atau kelas tertentu.
3. Fleksibel untuk memilih *fitur* dari *internal node* yang berbeda, *fitur* yang terpilih akan membedakan suatu kriteria dibandingkan kriteria yang lain dalam *node* yang sama. Kefleksibelan metode pohon keputusan ini meningkatkan kualitas keputusan yang dihasilkan jika dibandingkan ketika menggunakan metode penghitungan satu tahap yang lebih konvensional
4. Dalam analisis *multivariant*, dengan kriteria dan kelas yang jumlahnya sangat banyak, seorang penguji biasanya perlu untuk mengestimasikan baik itu

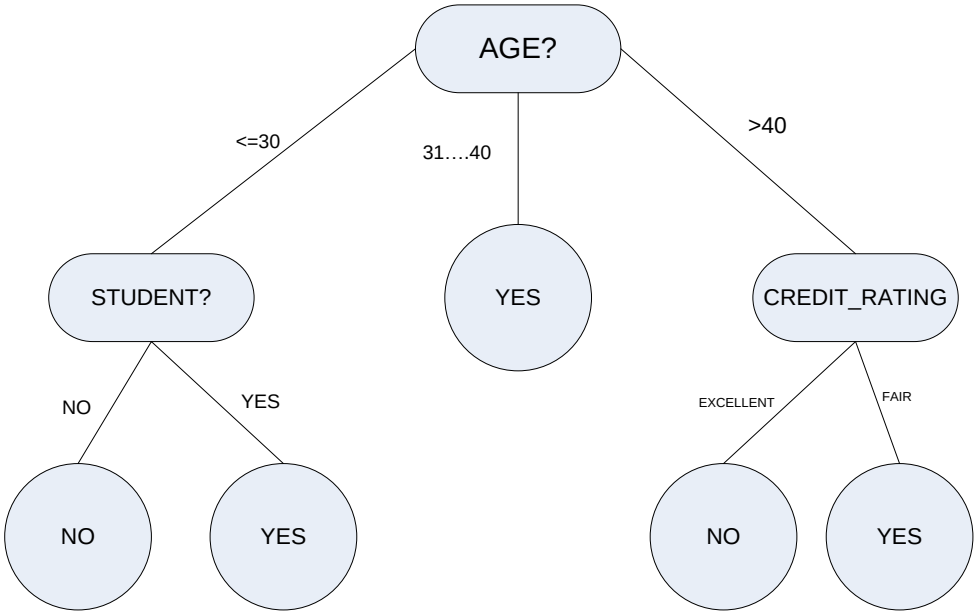
distribusi dimensi tinggi ataupun parameter tertentu dari distribusi kelas tersebut. Metode pohon keputusan dapat menghindari munculnya permasalahan ini dengan menggunakan kriteria yang jumlahnya lebih sedikit pada setiap *node* internal tanpa banyak mengurangi kualitas keputusan yang dihasilkan.

Sedangkan kekurangan dari pohon keputusan adalah :

1. Terjadi *overlap* terutama ketika kelas-kelas dan kriteria yang digunakan jumlahnya sangat banyak. Hal tersebut juga dapat menyebabkan meningkatnya waktu pengambilan keputusan dan jumlah memori yang diperlukan.
2. Pengakumulasian jumlah *error* dari setiap tingkat dalam sebuah pohon keputusan yang besar.
3. Kesulitan dalam mendesain pohon keputusan yang optimal.
4. Hasil kualitas keputusan yang didapatkan dari metode pohon keputusan sangat tergantung pada bagaimana pohon tersebut didesain.

Bagian awal dari pohon keputusan ini adalah titik akar (*root*), sedangkan setiap cabang dari *decision tree* merupakan pembagian berdasarkan hasil uji, dan titik akhir (*leaf*) merupakan pembagian kelas yang dihasilkan. Pada umumnya proses dari sistem *decision tree* adalah mengadopsi strategi pencarian *top-down* untuk solusi ruang pencariannya. Pada proses mengklasifikasikan sampel yang tidak diketahui, nilai atribut akan diuji pada *decision tree* dengan cara melacak jalur dari titik akar sampai titik akhir, kemudian akan diprediksikan kelas yang ditempati sampel baru tersebut. *Decision tree* mempunyai 3 tipe simpul yaitu:

1. Simpul akar dimana tidak memiliki cabang yang masuk dan memiliki cabang lebih dari satu, terkadang tidak memiliki cabang sama sekali.
2. Simpul internal dimana hanya memiliki 1 cabang yang masuk, dan memiliki lebih dari 1 cabang yang keluar.
3. Simpul daun atau simpul akhir dimana hanya memiliki 1 cabang yang masuk, dan tidak memiliki cabang sama sekali dan menandai bahwa simpul tersebut merupakan label kelas.



Gambar 2.1: Model Pohon Keputusan

Disini setiap percabangan menyatakan kondisi yang harus dipenuhi dan tiap ujung pohon menyatakan kelas data. Setelah sebuah pohon keputusan dibangun maka dapat digunakan untuk mengklasifikasikan *record* yang belum ada kelasnya. Dimulai dari *node root*, menggunakan tes terhadap atribut dari *record* yang belum ada kelasnya tersebut lalu mengikuti cabang yang sesuai dengan hasil dari tes tersebut, yang akan membawa kepada *internal node* (*node* yang memiliki satu cabang masuk dan dua atau lebih cabang yang keluar), dengan cara harus melakukan tes lagi terhadap atribut atau *node* daun. *Record* yang kelasnya tidak diketahui kemudian diberikan kelas yang sesuai dengan kelas yang ada pada *node* daun. Pada pohon keputusan setiap simpul daun menandai label kelas. Proses dalam pohon keputusan yaitu mengubah bentuk data (tabel) menjadi model pohon (*tree*) kemudian mengubah model pohon tersebut menjadi aturan (*rule*).

2.5 Algoritma C4.5

Algoritma C4.5 diperkenalkan oleh Quinlan (1996) sebagai versi perbaikan dari ID3. Dala ID3, induksi decision tree hanya bisa dilakukan pada fitur bertipe kategorikal (nominal atau ordinal), sedangan tipe numerik (interval atau rasio) tidak apat digunakan (Eko Prasetyo, 2013).

Yang menjadi hal penting dalam induksi decision tree adalah bagaimana menyatakan syarat pengujian pada node. Ada 3 kelompok penting dalam syarat pengujian node :

1. Fitur biner

   Adalah Fitur yang hanya mempunyai dua nilai berbeda. Syarat pengujian ketik fitur ini menjadi node (akar maupun interval) hanya punya dua pilihan cabang.
2. Fitur kategorikal

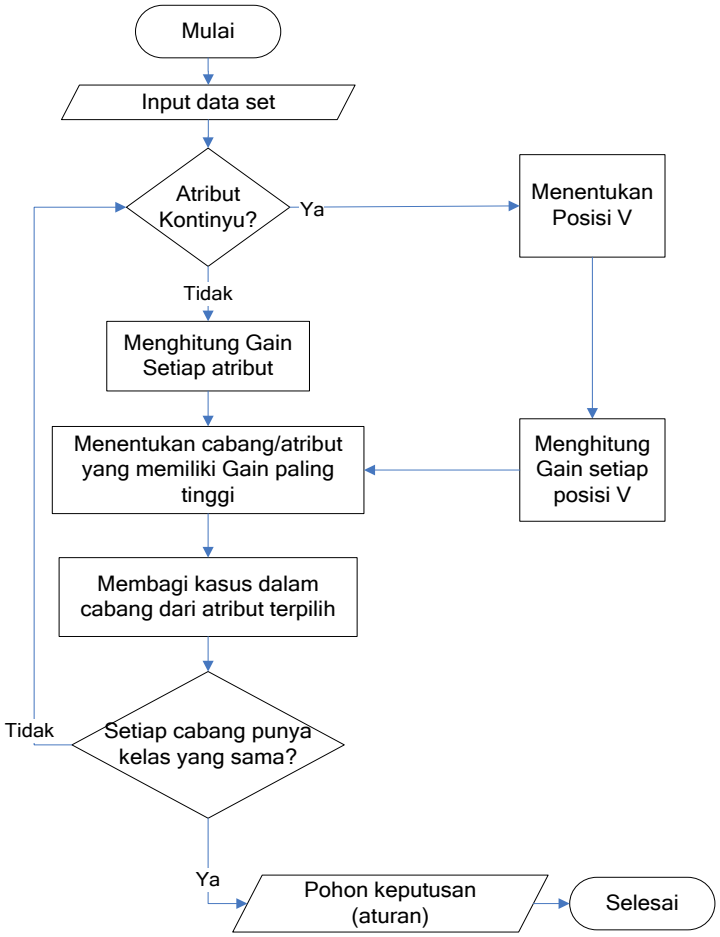
   Untuk fitur yang nilainya bertipe kategorikal (nominal atau ordinal) bisa mempunyai beberapa nilai berbeda. Secara umum ada 2 pemecahan yaitu pemecahan biner (*binary splitting*) dan (*multi splitting*).
3. Fitur numerik

   Untuk fitur bertipe numerik, Syarat pengujian dalam node (akar maupun internal) dinyatakan dengan pengujian perbandingan ($A \leq V$) atau ($A > V$) dengan hasil biner.

Secara umum algoritma C4.5 untuk membangun pohon keputusan adalah sebagai berikut:

1. Pilih atribut sebagai akar.
2. Buat cabang untuk tiap-tiap nilai.
3. Bagi kasus dalam cabang.
4. Ulangi proses untuk setiap cabang sampai semua kasus pada cabang memiliki kelas yang sama.

Berikut ini akan dijelaskan secara lebih detail algoritma C4.5 menggunakan *flowcart* yang disajikan pada **gambar 2.2**



Gambar 2.2 *Flowchart* algoritma *Decision Tree C4.5*

Untuk memilih atribut sebagai simpul akar (*root node*) atau simpul dalam (*internal node*), didasarkan pada nilai *information gain* tertinggi dari atribut-atribut yang ada. Sebelum perhitungan *information gain*, akan dilakukan perhitungan *entropy*. *Entropy* digunakan untuk menentukan node selanjutnya yang akan menjadi pemecah data latih selanjutnya untuk mengukur tingkat homogenitas distribusi kelas dari sebuah himpunan data (*data set*). Semakin tinggi tingkat *entropy* dari sebuah data maka semakin homogen distribusi kelas pada data tersebut. Perhitungan *information gain* menggunakan rumus 2.2, sedangkan *entropy* menggunakan rumus 2.3.

$$Gain(S,A) = Entropy\ (S) - \sum_{i=1}^{n} \frac{|S_i|}{|S|} * Entropy\ (S_i) \quad (2.2)$$

dimana,

S : Himpunan kasus

A : Atribut

n : Jumlah partisi atribut A

|Si| : Jumlah kasus pada partisi ke i

|S| : Jumlah kasus dalam S

$$Entropy(S) = -\sum_{i=1}^{n} p_i * log_2 p_i \quad (2.3)$$

dimana,

S : Himpunan kasus

A : Fitur

n : Jumlah partisi S

p_i : Proporsi dari S_i terhadap S

Selain *Information Gain* kriteria yang lain untuk memilih atribut sebagai pemecah adalah *Rasio Gain*. Rasio gain sendiri digunakan untuk memperbaiki information gain dengan menggunakan rumus menggunakan rumus 2.4, sedangkan *SplitInfomation* adalah normalisasi pada information gain untuk menghitung rasio perolehan suatu term baru sebagai pemisahan informasi nilai split information pada atribut A yang didapatkan dengan menggunakan rumus 2.5.

$$GainRasio\ (S,A) = \frac{Gain\ (S,A)}{SplitInformation\ \ (S,A)} \quad (2.4)$$

$$SplitInformation(S,A) = -\sum_{i=1}^{c} \frac{S_i}{S} log_2 \frac{S_i}{S} \quad (2.5)$$

dimana S_1 sampai S_c adalah c subset yang dihasilkan dari pemecahan S dengan menggunakan atribut A yang mempunyai sebanyak c nilai.

2.5.1 Contoh Perhitungan

Berikut ini akan dijelaskan ilustrasi dari alur proses perhitungan algoritma *Decision Tree C4.5*. Data set yang digunakan pada contoh ini adalah

data untuk menentukan *Yes* atau *No* dengan beberapa atribut yaitu atribut *outlook, temperature*, *humidity*, dan *windy*. Dimana atribut *temperature* dan *humidity* bertipe kontinyu sedangkan *outlook* dan *windy* bertipe kategorikal. Sedangkan kolom *Yes* adalah kelas tujuannya atau label kelas-nya.

Tabel 2.1 Contoh data set

Outlook	Temperature	Humidity	Windy	play
Sunny	75	70	TRUE	Yes
Sunny	80	90	TRUE	No
Sunny	85	85	FALSE	No
Sunny	72	95	FALSE	No
Sunny	69	70	FALSE	Yes
Overcast	72	90	TRUE	Yes
Overcast	83	78	FALSE	Yes
Overcast	64	65	TRUE	Yes
Overcast	81	75	FALSE	Yes
Rain	71	80	TRUE	No
Rain	65	70	TRUE	No
Rain	75	80	FALSE	Yes
Rain	68	80	FALSE	Yes
Rain	70	96	FALSE	Yes

Pada contoh ini rumus yang digunakan untuk memilih atribut sebagai *node* adalah rumus *information gain*. Proses pertama adalah menghitung *entropy* untuk semua data.

Jumlah play yes = 9

Jumlah play no = 5

Berikut adalah perhitungan *entropy* untuk semua data:

$$Entropy(S) = -\frac{9}{14} * log_2\left(\frac{9}{14}\right) - \frac{5}{14} * log_2\left(\frac{5}{14}\right)$$

$$= 0.9403$$

Selanjutnya menghitung *gain* untuk setiap atribut. Berikut adalah contoh perhitungan gain untuk atribut *outlook*:

Tabel 2.2 Distribusi jumlah atribut *outlook*

Nilai Outlook	∑ Yes	∑ No	Total
Sunny	2	3	5
Overcast	4	0	4
Rain	3	2	5

Berdasarkan tabel 2.2, maka nilai *information gain* untuk atribut *outlook* adalah sebagai berikut:

$$Gain(outlook)$$

$$= 0.9403 - \left(\frac{5}{14} * \left(-\frac{2}{5} * log_2\left(\frac{2}{5}\right) - \frac{3}{5} * log_2\left(\frac{3}{5}\right)\right)\right.$$
$$+ \frac{4}{14} * \left(-\frac{4}{4} * log_2\left(\frac{4}{4}\right) - \frac{0}{4} * log_2\left(\frac{0}{4}\right)\right)$$
$$\left. + \frac{5}{14} * \left(-\frac{3}{5} * log_2\left(\frac{3}{5}\right) - \frac{2}{5} * log_2\left(\frac{2}{5}\right)\right)\right)$$
$$= 0.9403 - 0.6936$$
$$= 0.2467$$

Untuk perhitungan atribut yang bertipe kontinyu, harus menentukan *posisi V* terbaik yang dinyatakan dalam perbandingan ($A \leq V$) atau ($A > V$). Berikut akan dijelaskan contoh perhitungan dari atribut *temperature.*

Misal posisi V yang akan digunakan pada atribut *temperature* adalah 70,75,dan 80, kemudian dihitung nilai *information gain*-nya.

Contoh perhitungan *temperature* posisi v=70:

$$Gain(temperature)$$

$$= 0.9403 - \left(\frac{2}{14} * \left(-\frac{4}{2} * log_2\left(\frac{4}{2}\right) - \frac{1}{2} * log_2\left(\frac{1}{2}\right)\right)\right.$$
$$\left. + \frac{12}{14} * \left(-\frac{5}{12} * log_2\left(\frac{5}{12}\right) - \frac{4}{12} * log_2\left(\frac{4}{12}\right)\right)\right)$$
$$= 0.9403 - 0.895$$
$$= 0.0453$$

Berikut hasil perhitungan atribut numerik untuk setiap posisi yang telah ditentukan:

Tabel 2.3 Jumlah dan hasil gain posisi V untuk atribut *temperature*

Temperature	**70**		**75**		**80**	
	≤	>	≤	>	≤	>
Yes	4	5	7	2	7	2
No	1	4	3	2	4	1
Jumlah	**5**	**9**	**10**	**4**	**11**	**3**
Entropy	**0.722**	**0.991**	**0.881**	**1.000**	**0.946**	**0.918**
Gain	0.0453		0.0251		0.0005	

Berdasarkan tabel 2.3, nilai gain tertinggi adalah didapatkan pada v=70, oleh karena itu, untuk fitur *temperature* dilakukan diskretisasi pada v=70 ketika menghitung entropy dan gain pada semua fitur. Hasil perhitungan pada setiap atribut disajikan pada tabel 2.4

Tabel 2.4 Jumlah dan hasil gain posisi V untuk atribut *humidity*

humidity	**70**		**75**		**80**		**85**	
	≤	>	≤	>	≤	>	≤	>
Yes	2	7	3	6	7	2	7	2
No	1	4	1	4	2	3	3	2
Jumlah	**3**	**11**	**4**	**10**	**9**	**5**	**10**	**4**
Entropy	**0.9183**	**0.9457**	**0.8113**	**0.9710**	**0.7624**	**0.9710**	**0.8813**	**1.000**
Gain	0.0005		0.0150		0.1022		0.0251	

Berdasarkan tabel 2.4, nilai gain tertinggi adalah didapatkan pada v=80, oleh karena itu, untuk fitur *humidity* dilakukan diskretisasi pada v=80 ketika menghitung entropy dan gain pada semua fitur. Hasil perhitungan pada setiap atribut disajikan pada tabel 2.5

Tabel 2.5 Hasil perhitungan *Information gain* untuk setiap atribut

		Jumlah	**Yes**	**No**	**Entropy**	**Gain**
Total		14	9	5	0.9403	
Outlook	**Sunny**	5	2	3	0.9710	0.2467
	Overcast	4	4	0	0.0000	
	Rain	5	3	2	0.9710	
Temperature	**≤ 70**	5	4	1	0.7219	0.0453

	> 70	9	5	4	0.9911	
Humidity	**≤ 80**	9	7	2	0.7642	0.1022
	> 80	5	2	3	0.9710	
Windy	**TRUE**	6	3	3	1.0000	0.0481
	FALSE	8	6	2	0.8113	

Berdasarkan tabel 2.5 menunjukkan bahwa atribut *outlook* memiliki nilai gain tertinggi, maka atribut *outlook* akan menjadi *node*. Karena atribut outlook memliki tiga nilai atribut atau lebih dari dua, maka dilakukan perhitungan rasio gain untuk memilih pilihan percabangan terbaik. Berikut adalah contoh perhitungan rasio gain untuk pilihan percabangan {sunny, overcast, rain}.

$$Split\ info(Semua, overcast) = \left(-\frac{5}{14} * log_2\left(\frac{5}{14}\right)\right) + \left(-\frac{4}{14} * log_2\left(\frac{4}{14}\right)\right) + \left(-\frac{5}{14} * log_2\left(\frac{5}{14}\right)\right)$$

$$= 0.531 + 0.516 + 0.531 = 1.5774$$

$$Rasio\ Gain(Semua, overcast) = \frac{0.2467}{1.5774} = 0.16$$

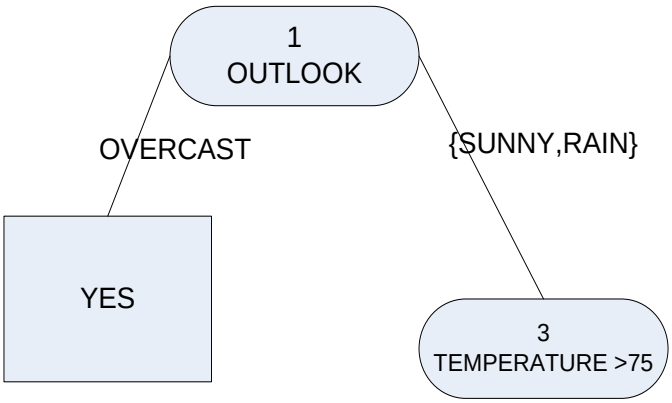
Hasil untuk perhitungan *rasio gain* lainnya ada pada tabel 2.6.

Tabel 2.6 Hasil perhitungan *Rasio gain* untuk setiap pilihan cabang

		Jumlah	**Split Inf**	**Gain**	**Rasio Gain**
Total		14		0.2467	
Pilihan 1	**sunny**	5	1.5774		0.16
	overcast	4			
	rain	5			
Pilihan 2	**sunny**	5	0.9403		0.26
	overcast Rain	9			
Pilihan 3	**sunny overcast**	9	0.9403		0.26
	rain	5			
Pilihan 4	**sunny Rain**	10	0.8631		0.29
	overcast	4			

Dari tabel 2.5 pilihan 4 yaitu {*sunny, rain*} dan {*overcast*} memiliki nilai *rasio gain* tertinggi, maka atribut terpilih (*outlook*) akan dibagi menjadi dua cabang. Data pada kasus pembagian OVERCAST memiliki kelas yang sama maka node

ini akan menjadi daun (*leaf*). Sedangkan node untuk kasus SUNNY dan RAIN masih ada kelas yang tidak sama, maka node ini akan memilih atribut sebagai pemecah, seperti ditunjukkan pada gambar 2.3. Pembagian cabang disajikan pada tabel 2.7 dan tabel 2.8.



Gambar 2.3 Hasil pembentukan cabang pada node akar

Tabel 2.7 Data pada kasus outlook sunny dan rain

Outlook	Temperature	Humidity	Windy	Play
sunny	75	70	TRUE	Yes
sunny	80	90	TRUE	No
sunny	85	85	FALSE	No
sunny	72	95	FALSE	No
sunny	69	70	FALSE	Yes
rain	71	80	TRUE	No
rain	65	70	TRUE	No
rain	75	80	FALSE	Yes
rain	68	80	FALSE	Yes
rain	70	96	FALSE	Yes

Tabel 2.8 pemisahan data pada kasus outlook overcast

Outlook	Temperature	Humidity	Windy	Play
overcast	72	90	TRUE	Yes
overcast	83	78	FALSE	Yes
overcast	64	65	TRUE	Yes
overcast	81	75	FALSE	Yes

Selanjutnya, memilih atribut kembali untuk atribut ‘temperature, posisi V yang digunakan adalah 75. Oleh karena itu, untuk fitur *temperature* dilakukan diskretisasi pada v=75 ketika menghitung entropy dan gain pada semua fitur. Hasil uji coba pada fitur ‘temperature dengan menghitung nilai gainnya disajikan pada tabel 2.9.

Tabel 2.9 Jumlah dan hasil gain posisi V untuk atribut *temperature*

Temperature	70		75		80	
	≤	>	≤	>	≤	>
Yes	3	2	5	0	5	0
No	1	4	3	2	4	1
Jumlah	4	6	8	2	9	1
Entropy	0.813	0.9183	0.9544	0.000	0.9911	0.000
Gain	0,1245		0,2365		0,1080	

Hasil uji coba pada fitur ‘humidity’ dengan menggunakan nilai gain disajikan pada tabel 2.10. nilai gain tertinggi didapatkan pada posisi v=80. Oleh karena itu, untuk fitur ‘humidity’ dilakukan diskretasi pada v=80 ketika menghitung entropy dan gain pada semua fitur.

Tabel 2.10 Jumlah dan hasil gain posisi V untuk atribut *Humidity*

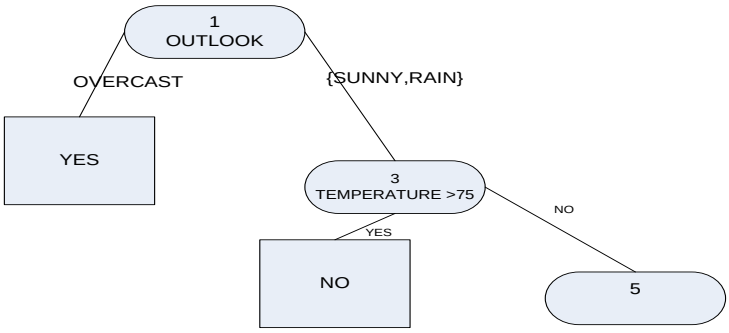
humidity	70		75		80		85	
	≤	>	≤	>	≤	>	≤	>
Yes	2	3	2	3	4	3	4	1
No	1	4	1	4	2	1	3	2
jumlah	3	7	3	7	6	4	7	3
Entropy	0.9183	0.9852	0.9183	0.9852	0.9183	0.8113	0.9852	0.9183
Gain	0.0349		0.0349		0.1245		0.0349	

Selanjutnya dihitung entropy untuk setiap fitur kelas, kemudian dihitung gain untuk setiap fitur, hasilnya disajikan pada tabel 2.11

Tabel 2.11 Hasil perhitungan *Information gain* untuk setiap atribut

		Jumlah	Yes	No	Entropy	Gain
Total		10	5	5	1.000	
Outlook	**Sunny**	5	2	3	0.9710	0.0290
	Rain	5	3	2	0.9710	
Temperature	**≤ 75**	8	5	3	0.9554	0.2365
	> 75	2	0	2	0	
Humidity	**≤ 80**	6	4	2	0.9183	0.1245
	> 80	4	1	3	0.8113	
Windy	**TRUE**	6	4	2	0.9183	0.1245
	FALSE	4	1	3	0.8113	

Berdasarkan tabel 2.11 menunjukkan bahwa atribut *temperature* memiliki nilai gain tertinggi, maka atribut *temperture* akan menjadi syarat kondisi *node*. Karena atribut *temperature* hanya memliki dua nilai atribut, maka tidak perlu dilakukan perhitungan rasio gain. Sehingga fitur *temperature* dijadikan syarat kondisi seperti ditunjukkan pada gambar 2.4. Pembagian cabang disajikan pada tabel 2.12 dan tabel 2.13.



Gambar 2.4 Hasil pembentukan cabang pada node 3

Tabel 2.12 data pada kasus temperature >75

Outlook	Temperature	Humidity	Windy	Play
sunny	75	70	TRUE	Yes
sunny	72	95	FALSE	No
sunny	69	70	FALSE	Yes
rain	71	80	TRUE	No

rain	65	70	TRUE	No
rain	75	80	FALSE	Yes
rain	68	80	FALSE	Yes
rain	70	96	FALSE	Yes

Tabel 2.13 pemisahan data pada kasus temperature >75

Outlook	**Temperature**	**Humidity**	**Windy**	**Play**
sunny	80	90	TRUE	No
sunny	85	85	FALSE	No

Selanjutnya, memilih atribut kembali untuk atribut ‘temperature, posisi V yang digunakan adalah 70. Oleh karena itu, untuk fitur ‘temperature dilakukan diskretisasi pada v=70 ketika menghitung entropy dan gain pada semua fitur. Hasil uji coba pada fitur ‘temperature’ dengan menghitung nilai gainnya disajikan pada tabel 2.14.

Tabel 2.14 Jumlah dan hasil gain posisi V untuk atribut *temperature*

Temperature	70		75	
	≤	>	≤	>
Yes	3	2	5	0
No	1	2	3	0
Jumlah	4	4	8	0
Entropy	0,8113	1.000	0.9554	0.000
Gain	0,0488		0	

Hasil uji coba pada fitur ‘humidity’ dengan menggunakan nilai gain disajikan pada tabel 2.15. nilai gain tertinggi didapatkan pada posisi v=80. Oleh karena itu, untuk fitur ‘humidity’ dilakukan diskretasi pada v=80 ketika menghitung entropy dan gain pada semua fitur.

Tabel 2.15 Jumlah dan hasil gain posisi V untuk atribut *humidity*

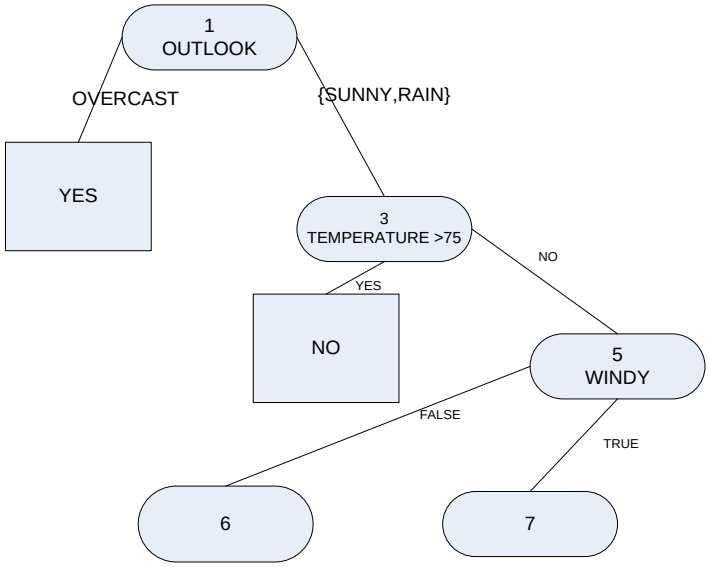
humidity	70		75		80	
	≤	>	≤	>	≤	>
Yes	2	3	2	3	4	1
No	1	2	1	2	2	1
Jumlah	3	5	3	5	6	2
Entropy	0.9183	0.9710	0.9183	0.9710	0.9183	1.000
Gain	0,0032		0,0032		0,0157	

Selanjutnya dihitung entropy untuk setiap fitur kelas, kemudian dihitung gain untuk setiap fitur, hasilnya disajikan pada tabel 2.16

Tabel 2.16 Hasil perhitungan *Information gain* untuk setiap atribut

		Jumlah	**Yes**	**No**	**Entropy**	**Gain**
Total		8	5	3	0,9544	
Outlook	**Sunny**	3	2	1	0.9183	0.0032
	Rain	5	3	2	0.9710	
Temperature	**≤ 75**	4	3	1	0.8113	0.0488
	> 75	4	2	2	1.0000	
Humidity	**≤ 80**	6	4	2	0.9183	0.0157
	> 80	2	1	1	1.0000	
Windy	**TRUE**	5	4	1	0.7219	0.1589
	FALSE	3	1	2	0.9813	

Berdasarkan tabel 2.15 menunjukkan bahwa atribut *windy* memiliki nilai gain tertinggi, maka atribut *windy* akan menjadi *node*. Karena atribut *windy* hanya memliki dua nilai atribut, maka tidak perlu dilakukan perhitungan rasio gain. Sehingga fitur *windy* dijadikan syarat kondisi seperti ditunjukkan pada gambar 2.5. Pembagian cabang disajikan pada tabel 2.17 dan tabel 2.18.



Gambar 2.5 Hasil pembentukan cabang pada node 5

Tabel 2.17 data pada kasus windy

Outlook	Temperature	Humidity	Windy	Play
rain	70	96	FALSE	Yes
sunny	72	95	FALSE	No
sunny	69	70	FALSE	Yes
rain	75	80	FALSE	Yes
rain	68	80	FALSE	Yes

Tabel 2.18 pemisahan data pada kasus windy

Outlook	Temperature	Humidity	Windy	Play
sunny	75	70	TRUE	Yes
rain	71	80	TRUE	No
rain	65	70	TRUE	No

Selanjutnya, memilih atribut kembali untuk atribut ‘temperature, posisi V yang digunakan adalah 70. Oleh karena itu, untuk fitur ‘temperature dilakukan diskretisasi pada v=70 ketika menghitung entropy dan gain pada semua fitur. Hasil uji coba pada fitur ‘temperature’ dengan menghitung nilai gainnya disajikan pada tabel 2.19.

Tabel 2.19 Jumlah dan hasil gain posisi V untuk atribut *temperature*

Temperature	70		75	
	≤	>	≤	>
Yes	3	1	3	1
No	0	1	1	0
Jumlah	3	2	4	1
Entropy	0.0000	1.000	0,8113	0.0000
Gain	0,3219		0.0729	

Hasil uji coba pada fitur 'humidity' dengan menggunakan nilai gain disajikan pada tabel 2.20. nilai gain tertinggi didapatkan pada posisi v=80. Oleh karena itu, untuk fitur 'humidity' dilakukan diskretasi pada v=80 ketika menghitung entropy dan gain pada semua fitur.

Tabel 2.20 Jumlah dan hasil gain posisi V untuk atribut *humidity*

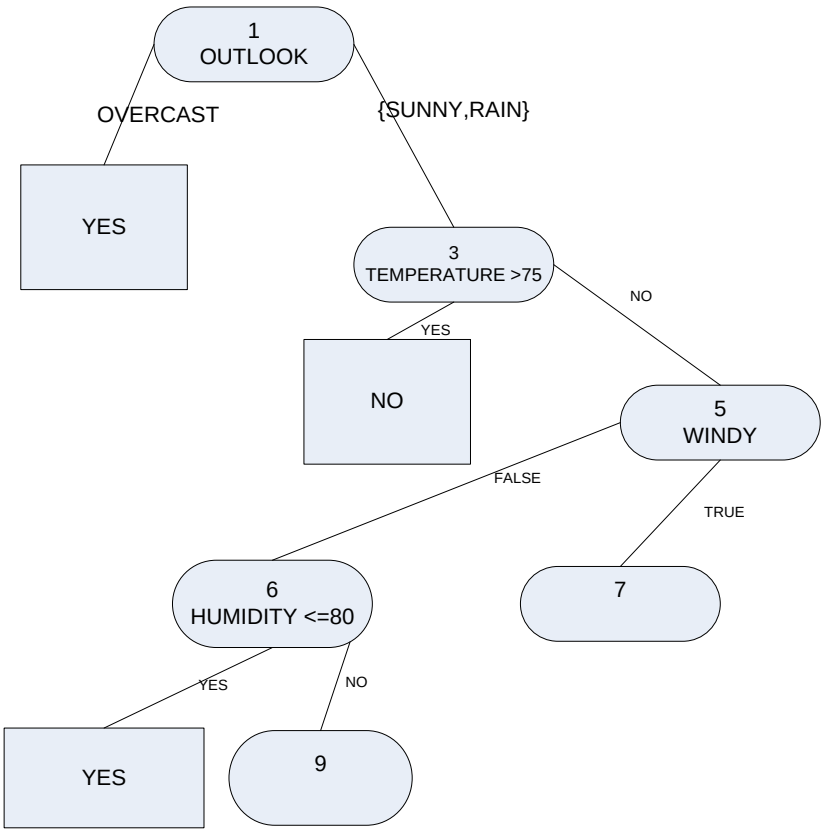
humidity	75		80		85	
	≤	>	≤	>	≤	>
Yes	1	3	3	1	4	1
No	0	1	0	1	2	1
Jumlah	1	4	3	2	6	2
Entropy	0.0000	0.8113	0.0000	1.000	0.9183	0.0000
Gain	0,0729		0,3219		0,3219	

Selanjutnya dihitung entropy untuk setiap fitur kelas, kemudian dihitung gain untuk setiap fitur, hasilnya disajikan pada tabel 2.21

Tabel 2.21 Hasil perhitungan *Information gain* untuk setiap atribut

		Jumlah	**Yes**	**No**	**Entropy**	**Gain**
Total		5	4	1	0,7219	
Outlook	**Sunny**	2	1	1	1.000	0.3219
	Rain	3	3	0	0.000	
Temperature	**≤ 70**	3	3	0	0.000	0.3219
	> 70	4	2	2	1.000	
Humidity	**≤ 80**	3	3	0	0.000	0.3219
	> 80	2	1	1	1.000	

Berdasarkan tabel 2.21 menunjukkan bahwa ketiga atribut memiliki nilai gain tertinggi yang sama, maka atribut *humidity* akan dijadikan syarat kondisi di node 6. seperti ditunjukkan pada gambar 2.6. Pembagian cabang disajikan pada tabel 2.22 dan tabel 2.23.



Gambar 2.6 Hasil pembentukan cabang pada node 6

Tabel 2.22 data pada kasus humidity

Outlook	Temperature	Humidity	Windy	Play
sunny	69	70	FALSE	Yes
rain	75	80	FALSE	Yes
rain	68	80	FALSE	Yes

Tabel 2.23 pembagian data pada kasus humidity

Outlook	Temperature	Humidity	Windy	Play
rain	70	96	FALSE	Yes
sunny	72	95	FALSE	No

Selanjutnya, memilih atribut kembali untuk atribut ‘temperature, posisi V yang digunakan adalah 70. Oleh karena itu, untuk fitur ‘temperature dilakukan diskretisasi pada v=70 ketika menghitung entropy dan gain pada semua fitur. Hasil uji coba pada fitur ‘temperature’ dengan menghitung nilai gainnya disajikan pada tabel 2.24

Tabel 2.24 Jumlah dan hasil gain posisi V untuk atribut *temperature*

Temperature	70		75	
	≤	>	≤	>
Yes	0	1	1	0
No	1	1	2	0
Jumlah	1	2	3	0
Entropy	0.0000	1.0000	0.9183	0.0000
Gain	0,2516		0.0000	

Hasil uji coba pada fitur ‘humidity’ dengan menggunakan nilai gain disajikan pada tabel 2.25. nilai gain hanya didapatkan pada posisi v=75. Oleh karena itu, untuk fitur ‘humidity’ dilakukan diskretasi pada v=75 ketika menghitung entropy dan gain pada semua fitur.

Tabel 2.25 Jumlah dan hasil gain posisi V untuk atribut *humidity*

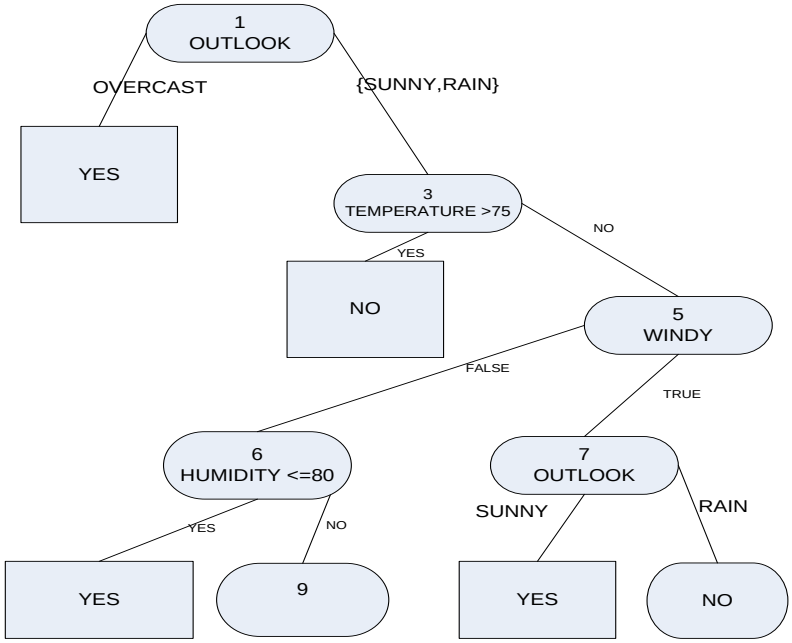
humidity	75	
	≤	≤
Yes	1	0
No	1	1
Jumlah	2	1
Entropy	1.0000	0.0000
Gain	0,2516	

Selanjutnya dihitung entropy untuk setiap fitur kelas, kemudian dihitung gain untuk setiap fitur, hasilnya disajikan pada tabel 2.26

Tabel 2.26 Hasil perhitungan *Information gain* untuk setiap atribut

		Jumlah	Yes	No	Entropy	Gain
Total		3	1	2	0,9183	
Outlook	**Sunny**	1	1	0	0.000	0.9183
	Rain	2	0	2	0.000	
Temperature	**≤ 70**	1	0	1	0.000	0.2516
	> 70	2	1	1	1.000	
Humidity	**≤ 75**	2	1	1	1.000	0.2516
	> 75	1	0	1	0.000	

Berdasarkan tabel 2.26 menunjukkan bahwa atribut outlook memiliki nilai gain tertinggi, maka atribut *outlook* akan dijadikan syarat kondisi di node 7. seperti ditunjukkan pada gambar 2.7. Pembagian cabang disajikan pada tabel 2.27 dan tabel 2.28.



Gambar 2.7 Hasil pembentukan cabang pada node 7

Tabel 2.27 Data pada kasus outlook

Outlook	Temperature	Humidity	Windy	Play
rain	71	80	TRUE	No
rain	65	70	TRUE	No

Tabel 2.28 Pembagian data pada kasus outlook

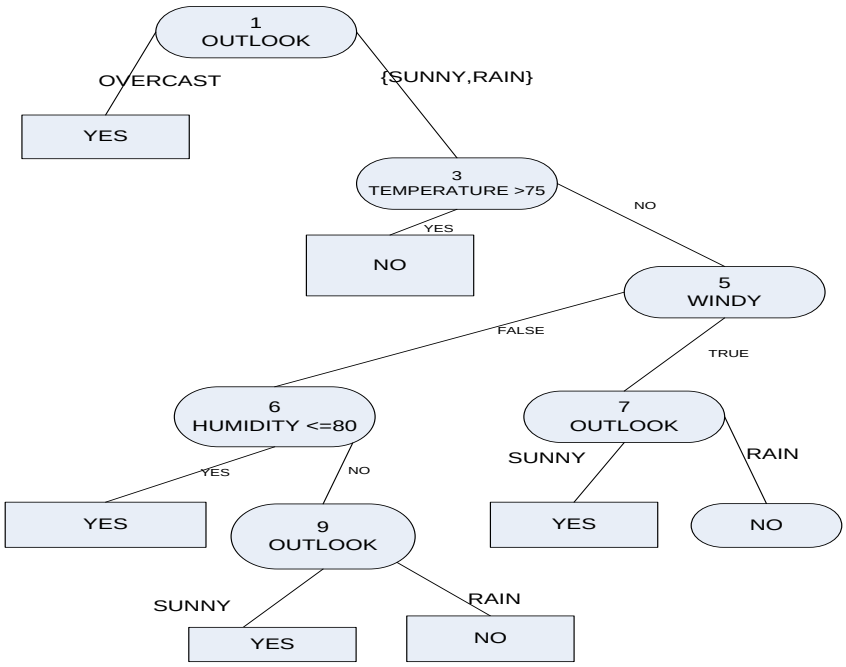
Outlook	Temperature	Humidity	Windy	Play
sunny	75	70	TRUE	Yes

Hanya tersisa 2 data dengan komposisi label 1 kelas Yes dan No. Nilai gain yang di dapat dari fitur *outlook,temperature,* dan *humidity* dipastikan juga sama sehingga bisa langsung dipilih salah satu, misalnya *outlook*. Hasil pohon yang dapat disajikan seperti gambar 2.8 Sebenarnya menjadi node cabang yang seharusnya tidak perlu dibentuk karena dapat menyebabkan pohon menjadi *overfitting*, di mana hal ini dapat mempengaruhi trade off antara akurasi prediksi terhadap pelatihan yang dilakukan.

Karena tidak ada lagi node yang harus diproses, maka induksi decision tree dinyatakan selesai. Hasil akhir dari decision tree disajikan seperti gambar 2.8.

Bentuk aturan IF THEN untuk decision tree sebagai berikut:

- IF outlook = overcast THEN play = yes
- IF outlook = {sunny,rain} AND temperature >75 THEN play = no
- IF outlook = {cerah,hujan} AND temperature <=75 AND windy = FALSE AND humidity <=80 THEN play = yes
- IF outlook = sunny AND temperature <=75 AND windy = FALSE AND humidity >80 THEN play= yes
- IF outlook = rain AND temperature <=75 AND windy = FALSE AND humidity >80 THEN play= no
- IF outlook = sunny AND humidity <=75 AND windy = TRUE THEN play = yes
- IF outlook = rain AND humidity <=75 AND windy = TRUE THEN play = no



Gambar 2.8 Hasil pembentukan cabang pada node 9

2.6 Penelitian Sebelumnya

Penelitian sebelumnya dilakukan Mohammad Luqman (2014) dengan judul Sistem Aplikasi Prediksi Lama Studi Mahasiswa Teknik Informatika Universitas Muhammadiyah Gresik Menggunakan Metode Fuzzy Inferensi Sugeno. Dengan menggunakan Metode Fuzzy Sugeno diharapkan dapat meningkatkan kinerja sistem dalam melakukan prediksi lama studi mahasiswa dengan menambahkan variabel sehingga sistem dapat bekerja dengan efektif dalam memprediksi lama studi. Dalam penelitian yang dilakukan proses pengujiannya menggunakan 80 data mahasiswa Teknik Informatika, dengan menggunakan factor nama, jarak tempuh, penghasilan orangtua, tanggungan orangtua, usia masuk dan nilai danem. Hasil dari pengujian system tersebut yang dilakukan pada 40 data mahasiswa sebagai *sample* untuk mengetahui seberapa besar tingkat kecenderungan penyelesaian studi mahasiswa, sistem memiliki nilai akurasi kebenaran sistem sebesar 72% dan nilai error sebesar 23%.

Penelitian selanjutnya dilakukan Setyowantono (2014) dengan judul Prediksi Penyelesaian Studi Mahasiswa Baru Dengan Metode Fuzzy Tsukamoto. Dengan menggunakan Metode Fuzzy Tsukamoto diharapkan dapat meningkatkan

kinerja sistem dalam melakukan prediksi lama studi mahasiswa dengan menambahkan variabel sehingga sistem dapat bekerja dengan efektif dalam memprediksi lama studi mahasiswa. Dalam penelitian yang dilakukan proses penggujiannya menggunakan 80 data mahasiswa teknik informatika dengan yang terdiri dari 40 mahasiswa telah lulus dan 40 mahasiswa belum lulus, dengan menggunakan factor nama, jarak tempuh, penghasilan orangtua, tanggungan orangtua, usia masuk dan nilai danem. Hasil dari penelitian tersebut dengan menggunakan metode Fuzzy Tsukamoto Tingkat validitas SPK dengan metode FIS Tsukamoto untuk menentukan prediksi lama studi mahasiswa baru termasuk dalam kategori baik. Hal ini ditunjukkan dengan hasil uji validitas SPK dengan membandingkan hasil perhitungan manual dan hasil SPK menggunakan 80 jenis data yang menghasilkan tingkat validitas SPK mencapai 72, 5% dan tingkat keerroran sebesar 18,8%.

Penelitian selanjutnya yang menggunakan metode pohon keputusan C4.5 adalah penelitian yang berjudul "*System prediksi prestasi akademik mahasiswa menggunakan metode decision tree C4.5 (Studi kasus:Jurusan Teknik informatika UNMUH GRESIK)"*, dibuat oleh Aunur Rasyid (Universitas Muhammadiyah Gresik, 2014). Tujuan dari penelitian tersebut adalah untuk menghasilkan informasi perkiraan kategori prestasi mahasiswa menggunakan metode *Decision Tree C4.5* sebagai peringatan dini dan motivasi mahasiswa dalam mendapatkan prestasi yang maksimal. Atribut-atribut yang digunakan adalah instansi sekolah asal (SMK,SMA atau MA), status sekolah asal (Negri atau Swasta), jurusan sekolah asal (IPA,IPS,Bahasa,Teknik,Administrasi), nilai rata-rata UN, status kerja (Sudah atau Belum), dan pihak yang mempengaruhi mahasiswa dalam memilih kuliah (Sendiri,Orang tua atau Orang lain). Hasil dari penelitian tentang system prediksi prestasi akademik mahasiswa yang dirancang menggunakan algoritma C4.5 mendapat tingkat akurasi tertinggi 90% dan tingkat keerorran 10% dari beberapa kali percobaan.